Space weather notes.

Background.

The goal of this project is to simulate individual particles in the magnetosphere. Delcourt wrote a program called code89t.f which simulates a single particle. code89t.f does not read in magnetic and electric fields, it calculates them. The magnetic field is calculated using the Tsyganenko model. For code89t.f the user specifies initial conditions for a single particle: energy, pitch, phase, magnetic local time, magnetic latitude, and distance from the earth's center. The code will then simulate the particle's movement in the magnetosphere and write out an output file with this information.

Modifications to code89t.f

Code89t.f was first modified to read in a grid along with magnetic and electric fields (so they don't need to be calculated anymore). The grid is called MHDGRID and specifies the positions of the electric and magnetic field vectors. The code performs interpolation to find vector values at positions in between grid points.

Originally code89t.f output data every delta T steps. The output consisted of energy, pitch, phase, magnetic latitude, magnetic local time, etc. This is fine for one particle but for many the output file was too large. In order to reduce the output file size the program was modified to calculate averages inside 1 RE cubes or "bins". The output is then average values inside a single bin, and there is one output line for each bin as the particle moves through space.

Code89t.f was again modified to compute initial conditions instead of specifying them. For each initial condition a range of values of is given, and the code will randomly pick values within each range. In this way the program can simulate thousands of particles in a single run. The initial conditions were polar wind, solar wind, auroral wind, plasmashphere, etc.

Finally, the code was parallelized to run in the cluster and modified to read V fields instead of E fields. E is then calculated as E=-V x B. After E is computed the V field is not used by the model anymore.

Cluster Fortran program.

First the code reads the initial parameters file with information about the kind of release (auroral, polar, plasmasphere, etc.), location of field files, etc.

Then two sets of input files are read: the velocity files and the magnetic field files. There are several sets of fields: SBz, Nov03, etc., each with a different number of files. These sets of files are independent of the release type, that is, one can have auroral release with SBz fields, auroral release with Nov03 fields, etc.

Each field file occurs at a given point in time; the times are given in the time file. The positions in space for the velocity and magnetic fields are given in the file MHDGRID. The code also reads the CAPS conditions, used to compute initial conditions for each particle.

The solar release case.

The solar wind case was the most difficult to implement. The problem is that most particles from the solar wind fly past the earth and generating many of them wastes disk space. It was decided to limit the number of particles that go through any bin to 100. After that the particle data is not written to disk for that bin. This process is performed in the Fortran cluster code.

IDL movie generation.

The program looks at the current directory to find out the release type. The IDL movie code reads the data from the cluster run, the time file, the B and other condition, and the CAPS conditions if needed. Its output are postscript files, one file for each frame and quantity computed (energy, pressure, density, etc.). If NUM_SUBIMAGES is greater than 1 then the program will generate intermediate frames by interpolating data from the calculated frames. The postscript frames are used by the "convert" command in math to generate mpeg files for each quantity.

Due to the time it takes to read the input files the program can restart at a given file input number. Look for variables numInputFiles, startWithFile, and fileSaveCount. As the program reads the data files it calculates density, energy, pressure, etc., for each point in a rectangular grid. The values calculated are for 1 RE^3 cubes. The dimensions of the computing grid are GSM X=[-70..30], Y=[-30..30], Z=[-30..30]. After fileSaveCount input data files the program saves the data[] array into an IDL data file using the save command. This is the file that is read in order to restart the program. If numInputFiles is equal to startWithFile then the data is read from the IDL saved file, and no more ascii data from the cluster is used. This option saves a lot time because reading the cluster ascii files takes several hours.

Note: There are two capabilities that we have not used for quite a while: generating velocity distribution plots (look for makeVelDistr flag) and making movies for the solar release case (look for SOLRE case).

Some IDL code parameters.

USE_PLSP_DATA. For plasmasphere release, set it to 1 to display data from 3Dplsp file.
USE_PARTICLE_DATA. For plasmasphere release, set it to 1 to display data from cluster run.
SKIP_NEGATIVE_START_TIMES. Set to 1 to skip data from negative times.
sliceTimeLenghMinutes. This is the frame length.

createAsciiDataFiles. Set it to 1 to save data in ASCII format.
MAX_SLICES. Maximum number of frames
sliceTimeLength. Time length of each frame in minutes.
NUM_SUBIMAGES. Used to smooth movies. Number of subframes that are generated by interpolating.
useVariableSaturation. Set it to 1 to make the saturation vary according to the counts in each bin.
debugParticles. Useful for debugging. Program stops after reading this number of particles.
makeVelDistro. Set it to 1 to generate velocity distribution plots. We have not generated these plots for a long time.
showQuantity array. Set elements to 1 to generate corresponding movie.
totalDialSets, dialMin, useBatchRanges. Used for velocity distribution plots.
minValue, maxValue. Determines plot data range in frames.